Project: Customer Relationship Management (CRM) System

1. Introduction

This document outlines the Low-Level Design (LLD) for a **Customer Relationship Management (CRM) System**, aimed at helping businesses manage interactions with customers, streamline processes, and improve profitability. The system supports customer data management, sales automation, customer support, and analytics to facilitate better customer engagement and decision-making.

This design supports both **Java (Spring Boot)** and **.NET (ASP.NET Core)** frameworks for backend development.

2. Module Overview

2.1 Customer Data Management Module

- Manages customer profiles, including contact details, purchase history, and communication logs.
- Provides tools for segmenting customers based on demographics and purchasing behavior.

2.2 Sales Automation Module

- Automates the sales pipeline from lead generation to deal closure.
- Tracks sales opportunities and activities, sending reminders and notifications.

2.3 Customer Support Module

- Manages customer service requests, including ticketing, complaint management, and resolution tracking.
- Provides a knowledge base for customers to find self-service solutions.

2.4 Marketing Automation Module

- Automates marketing campaigns through email, social media, and other digital platforms.
- Tracks campaign performance and customer responses.

2.5 Analytics and Reporting Module

- Generates reports on sales performance, customer behavior, and campaign effectiveness.
- Provides actionable insights for improving customer relationships and business performance.

3. Architecture Overview

3.1 Architectural Style

- **Frontend**: Angular or React for an interactive user interface.
- **Backend**: REST API-based architecture for managing customer data, sales activities, and reporting.
- **Database**: Relational Database (MySQL/PostgreSQL/SQL Server) for storing customer profiles, sales data, tickets, and campaign records.

3.2 Component Interaction

- The frontend interacts with the backend via REST APIs for managing customer data, tracking sales activities, and generating reports.
- The backend interfaces with the relational database for storing and retrieving data.

4. Module-Wise Design

4.1 Customer Data Management Module

4.1.1 Features

- Add, update, and view customer profiles.
- Manage customer interaction logs and history.
- Categorize customers based on attributes such as region, interests, and purchasing habits.

4.1.2 Data Flow

- Users interact with the frontend to add or modify customer information.
- The backend stores and retrieves data from the database.

4.1.3 Entities

- **CustomerProfile**
    - CustomerID
    - Name
    - ContactInfo
    - PurchaseHistory
    - SegmentationData

4.2 Sales Automation Module

4.2.1 Features

- Track leads, opportunities, and sales activities from initial contact to final deal closure.
- Automate follow-up reminders and notifications for sales teams.

4.2.2 Entities

- **SalesOpportunity**
    - OpportunityID
    - CustomerID
    - SalesStage
    - EstimatedValue
    - ClosingDate

4.3 Customer Support Module

4.3.1 Features

- Create and manage customer service tickets.
- Assign tickets to support agents, track ticket resolution status, and generate reports on issue resolution.

4.3.2 Entities

- **SupportTicket**
    - TicketID
    - CustomerID
    - IssueDescription
    - AssignedAgent
    - Status (Open/Closed)

4.4 Marketing Automation Module

4.4.1 Features

- Design and automate marketing campaigns (email, SMS, etc.).
- Track customer responses to campaigns and analyze effectiveness.

4.4.2 Entities

- **Campaign**

- CampaignID
- Name
- StartDate
- EndDate
- Type (Email/SMS/Social Media)

4.5 Analytics and Reporting Module

4.5.1 Features

- Generate reports and dashboards to evaluate sales performance, customer behavior, and marketing campaign outcomes.
- Offer insights for improving engagement and profitability.

4.5.2 Entities

- **Report**
  - ReportID
  - ReportType (Sales/Support/Marketing)
  - GeneratedDate
  - DataPoints

5. Deployment Strategy

5.1 Local Deployment

- Initial deployment on developer machines for testing and validation.

5.2 Staging and Production Environments

- Utilize cloud environments (AWS, Azure) for staging and production deployments, ensuring scalability and high availability.

6. Database Design

6.1 Tables and Relationships

- **CustomerProfile**: Primary Key: CustomerID.
- **SalesOpportunity**: Foreign Key: CustomerID.
- **SupportTicket**: Foreign Key: CustomerID.

- **Campaign**: Primary Key: CampaignID.
- **Report**: Foreign Key: ReportID.

7. User Interface Design

7.1 Wireframes

- **Dashboard**: Displays key metrics, sales opportunities, and support tickets.
- **Customer Management**: Interface for viewing and editing customer profiles.
- **Campaign Management**: Interface for creating and tracking marketing campaigns.

8. Non-Functional Requirements

8.1 Performance

- The system should be able to handle at least 5,000 active users simultaneously without performance degradation.
- Campaigns should be executed with minimal latency.

8.2 Usability

- The interface must be intuitive and accessible for sales, support, and marketing teams.
- Provide detailed help and tooltips for non-technical users.

8.3 Security

- Implement strong authentication and authorization mechanisms for different user roles.
- Encrypt sensitive customer data both at rest and in transit.

8.4 Scalability

- The CRM system should support scaling to accommodate additional modules, customers, and users as the business grows.

9. Assumptions and Constraints

9.1 Assumptions

- All users will have internet access for cloud-based CRM deployment.
- Customers will voluntarily provide feedback and participate in marketing campaigns.

9.2 Constraints

- The system will initially be rolled out for small and medium-sized businesses with up to 1,000 active users in the pilot phase.